Stream: Internet Engineering Task Force (IETF)
RFC: 0000
Category: Standards Track
Published: September 2021
ISSN: 2070-1721

Authors:
E. Birrane, *JHU/APL*
A. White, *JHU/APL*
S. Heiner, *JHU/APL*

RFC 0000
BPSec Default Security Contexts

Abstract

This document defines default integrity and confidentiality security contexts that can be used with the Bundle Protocol Security Protocol (BPSec) implementations. These security contexts are intended to be used for both testing the interoperability of BPSec implementations and for providing basic security operations when no other security contexts are defined or otherwise required for a network.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc0000.

Copyright Notice

Table of Contents

1. Introduction

The Bundle Protocol Security Protocol (BPSec) [RFCYYY2] specification provides inter-bundle integrity and confidentiality operations for networks deploying the Bundle Protocol (BP) [RFCYYY1]. BPSec defines BP extension blocks to carry security information produced under the auspices of some security context.

This document defines two security contexts (one for an integrity service and one for a confidentiality service) for populating BPSec Block Integrity Blocks (BIBs) and Block Confidentiality Blocks (BCBs). This document assumes familiarity with the concepts and terminology associated with BP and BPSec, as these security contexts are used with BPSec security blocks and other BP blocks carried within BP bundles.

These contexts generate information that **MUST** be encoded using the CBOR specification documented in [RFC8949].

2. Requirements Language

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Integrity Security Context BIB-HMAC-SHA2

3.1. Overview

The BIB-HMAC-SHA2 security context provides a keyed-hash Message Authentication Code (MAC) over a set of plain text information. This context uses the Secure Hash Algorithm 2 (SHA-2) discussed in [SHS] combined with the HMAC keyed hash discussed in [RFC2104]. The combination of HMAC and SHA-2 as the integrity mechanism for this security context was selected for two reasons:

1. The use of symmetric keys allows this security context to be used in places where an asymmetric-key infrastructure (such as a public key infrastructure) might be impractical.
2. The combination HMAC-SHA2 represents a well-supported and well-understood integrity mechanism with multiple implementations available.

BIB-HMAC-SHA2 supports three variants of HMAC-SHA, based on the supported length of the SHA-2 hash value. These variants correspond to "HMAC 256/256", "HMAC 384/384", and "HMAC 512/512" as defined in Table 7 ("HMAC Algorithm Values") of [RFC8152]. The selection of which variant is used by this context is provided as a security context parameter.

The output of the HMAC **MUST** be equal to the size of the SHA2 hashing function: 256 bits for SHA-256, 384 bits for SHA-384, and 512 bits for SHA-512.

The BIB-HMAC-SHA2 security context **MUST** have the security context identifier specified in Section 5.1.

3.2. Scope

The scope of BIB-HMAC-SHA2 is the set of information used to produce the plain text over which a keyed hash is calculated. This plain text is termed the "Integrity Protected Plain Text" (IPPT). The content of the IPPT is constructed as the concatenation of information whose integrity is being preserved from the BIB-HMAC-SHA2 security source to its security acceptor. There are five types of information that can be used in the generation of the IPPT, based on how broadly the concept of integrity is being applied. These five types of information, whether they are required, and why they are important for integrity, are discussed as follows.

Security target contents

The contents of the block-type-specific data field of the security target **MUST** be included in the IPPT. Including this information protects the security target data and is considered the minimal, required set of information for an integrity service on the security target.

IPPT Scope

The determination of which optional types of information were used when constructing the IPPT **MUST**, itself, always be included in the IPPT. Including this information ensures that the scope of the IPPT construction at a security source matches the scope of the IPPT construction at security verifiers and security acceptors.

Primary block

The primary block identifies a bundle and, once created, the contents of this block are immutable. Changes to the primary block associated with the security target indicate that the security target (and BIB) might no longer be in the correct bundle.

For example, if a security target and associated BIB are copied from one bundle to another bundle, the BIB might still contain a verifiable signature for the security target unless information associated with the bundle primary block is included in the keyed hash carried by the BIB.

Including this information in the IPPT protects the integrity of the association of the security target with a specific bundle.

Security target other fields

The other fields of the security target include block identification and processing information. Changing this information changes how the security target is treated by nodes in the network even when the "user data" of the security target are otherwise unchanged.

For example, if the block processing control flags of a security target are different at a security verifier than they were originally set at the security source then the policy for handling the security target has been modified.

Including this information in the IPPT protects the integrity of the policy and identification of the security target data.

BIB other fields

The other fields of the BIB include block identification and processing information. Changing this information changes how the BIB is treated by nodes in the network, even when other aspects of the BIB are unchanged.

For example, if the block processing control flags of the BIB are different at a security verifier than they were originally set at the security source, then the policy for handling the BIB has been modified.

Including this information in the IPPT protects the integrity of the policy and identification of the security service in the bundle.

> NOTE: The security context identifier and security context parameters of the security block are not included in the IPPT because these parameters, by definition, are required to verify or accept the security service. Successful verification at security verifiers and security acceptors implies that these parameters were unchanged since being specified at the security source. This is the case because keys cannot be re-used across security contexts, and because the integrity scope flags used to define the IPPT are included in the IPPT itself.

The scope of the BIB-HMAC-SHA2 security context is configured using an optional security context parameter.

3.3. Parameters

BIB-HMAC-SHA2 can be parameterized to select SHA-2 variants, communicate key information, and define the scope of the IPPT.

3.3.1. SHA Variant

This optional parameter identifies which variant of the SHA-2 algorithm is to be used in the generation of the authentication code.

This value **MUST** be encoded as a CBOR unsigned integer.

Valid values for this parameter are as follows.

Value	Description
5	HMAC 256/256 as defined in Table 7 ("HMAC Algorithm Values") of [RFC8152]
6	HMAC 384/384 as defined in Table 7 ("HMAC Algorithm Values") of [RFC8152]
7	HMAC 512/512 as defined in Table 7 ("HMAC Algorithm Values") of [RFC8152]

Table 1: SHA Variant Parameter Values

When not provided, implementations **SHOULD** assume a value of 6 (indicating use of HMAC 384/384), unless an alternate default is established by local security policy at the security source, verifiers, or acceptor of this integrity service.

3.3.2. Wrapped Key

This optional parameter contains the output of the AES key wrap authenticated encryption function (KW-AE) as defined in [RFC5649]. Specifically, this parameter holds the cipher text produced when running the KW-AE algorithm with the input string being the symmetric HMAC key used to generate the security results present in the security block. The value of this parameter is used as input to the AES key wrap authenticated decryption function (KW-AD) at security verifiers and security acceptors to determine the symmetric HMAC key needed for the proper validation of the security results in the security block.

This value **MUST** be encoded as a CBOR byte string.

If this parameter is not present then security verifiers and acceptors **MUST** determine the proper key as a function of their local BPSec policy and configuration.

3.3.3. Integrity Scope Flags

This optional parameter contains a series of flags that describe what information is to be included with the block-type-specific data when constructing the IPPT value.

This value **MUST** be represented as a CBOR unsigned integer, the value of which **MUST** be processed as a 16-bit field. The maximum value of this field, as a CBOR unsigned integer, **MUST** be 65535.

Implementations **MUST** set reserved and unassigned bits in this field to 0 when constructing these flags at a security source. Once set, the value of this field **MUST NOT** be altered until the security service is completed at the security acceptor in the network and removed from the bundle.

Bits in this field represent additional information to be included when generating an integrity signature over the security target. These bits are defined as follows.

Bit 0 (the low-order bit, 0x0001): Primary Block Flag

Bit 1 (0x0002): Target Header Flag

Bit 2 (0x0004): Security Header Flag

Bits 3-7: Reserved

Bits 8-15: Unassigned

3.3.4. Enumerations

The BIB-HMAC-SHA2 security context parameters are listed in Table 2. In this table, the "Parm Id" column refers to the expected Parameter Identifier described in Section 3.10 ("Parameter and Result Identification") of [RFCYYY2].

If the default value column is empty, this indicates that the security parameter does not have a default value.

Parm Id	Parm Name	CBOR Encoding Type	Default Value
1	SHA Variant	unsigned integer	6
2	Wrapped Key	Byte String	
3	Integrity Scope Flags	unsigned integer	7

Table 2: BIB-HMAC-SHA2 Security Parameters

3.4. Results

The BIB-HMAC-SHA2 security context results are listed in Table 3. In this table, the "Result Id" column refers to the expected Result Identifier described in Section 3.10 ("Parameter and Result Identification") of [RFCYYY2].

Result Id	Result Name	CBOR Encoding Type	Description
1	Expected HMAC	byte string	The output of the HMAC calculation at the security source.

Table 3: BIB-HMAC-SHA2 Security Results

3.5. Key Considerations

HMAC keys used with this context **MUST** be symmetric and **MUST** have a key length equal to the output of the HMAC. For this reason, HMAC key lengths will be integer divisible by 8 bytes and special padding-aware AES key wrap algorithms are not needed.

It is assumed that any security verifier or security acceptor performing an integrity verification can determine the proper HMAC key to be used. Potential sources of the HMAC key include (but are not limited to) the following:

- Pre-placed keys selected based on local policy.
- Keys extracted from material carried in the BIB.
- Session keys negotiated via a mechanism external to the BIB.

When an AES-KW wrapped key is present in a security block, it is assumed that security verifiers and security acceptors can independently determine the key encryption key (KEK) used in the wrapping of the symmetric HMAC key.

As discussed in Section 6 and emphasized here, it is strongly recommended that keys be protected once generated, both when they are stored and when they are transmitted.

3.6. Security Processing Considerations

An HMAC calculated over the same IPPT with the same key will always have the same value. This regularity can lead to practical side-channel attacks whereby an attacker could produce known plain text and a guess at an HMAC tag and observe the behavior of a verifier. With a modest number of trials, a side-channel attack could produce an HMAC tag for attacher-provided plain text without the attacker ever knowing the HMAC key.

A common method of observing the behavior of a verifier is precise analysis of the timing associated with comparisons. Therefore, one way to prevent behavior analysis of this type is to ensure that any comparisons of the supplied and expected authentication tag occur in constant time.

A constant-time comparison function **SHOULD** be used for the comparison of authentication tags by any implementation of this security context. In cases where such a function is difficult or impossible to use, the impact of side-channel (in general) and timing attacks (specifically) need to be considered as part of the implementation.

3.7. Canonicalization Algorithms

This section defines the canonicalization algorithm used to prepare the IPPT input to the BIB-HMAC-SHA2 integrity mechanism. The construction of the IPPT depends on the settings of the integrity scope flags that can be provided as part of customizing the behavior of this security context.

In all cases, the canonical form of any portion of an extension block **MUST** be performed as described in [RFCYYY2]. The canonicalization algorithms defined in [RFCYYY2] adhere to the canonical forms for extension blocks defined in [RFCYYY1] but resolve ambiguities related to how values are represented in CBOR.

The IPPT is constructed using the following process. While integrity scope flags might not be included in the BIB representing the security operation, they **MUST** be included in the IPPT value itself.

1. The canonical form of the IPPT starts as the CBOR encoding of the integrity scope flags in which all unset flags, reserved bits, and unassigned bits have been set to 0. For example, if the primary block flag, target header flag, and security header flag are each set, then the initial value of the canonical form of the IPPT will be 0x07.
2. If the primary block flag of the integrity scope flags is set to 1, then a canonical form of the bundle's primary block **MUST** be calculated and the result appended to the IPPT.
3. If the target header flag of the integrity scope flags is set to 1, then the canonical form of the block type code, block number, and block processing control flags associated with the security target **MUST** be calculated and, in that order, appended to the IPPT.
4. If the security header flag of the integrity scope flags is set to 1, then the canonical form of the block type code, block number, and block processing control flags associated with the BIB **MUST** be calculated and, in that order, appended to the IPPT.
5. The canonical form of the security target block-type-specific data **MUST** be calculated and appended to the IPPT.

3.8. Processing

3.8.1. Keyed Hash Generation

During keyed hash generation, two inputs are prepared for the the appropriate HMAC/SHA2 algorithm: the HMAC key and the IPPT. These data items **MUST** be generated as follows.

- The HMAC key **MUST** have the appropriate length as required by local security policy. The key can be generated specifically for this integrity service, given as part of local security policy, or through some other key management mechanism as discussed in Section 3.5.

- Prior to the generation of the IPPT, if a CRC value is present for the target block of the BIB, then that CRC value **MUST** be removed from the target block. This involves both removing the CRC value from the target block and setting the CRC Type field of the target block to "no CRC is present."
- Once CRC information is removed, the IPPT **MUST** be generated as discussed in Section 3.7.

Upon successful hash generation the following actions **MUST** occur.

- The keyed hash produced by the HMAC/SHA2 variant **MUST** be added as a security result for the BIB representing the security operation on this security target, as discussed in Section 3.4.

Finally, the BIB containing information about this security operation **MUST** be updated as follows. These operations can occur in any order.

- The security context identifier for the BIB **MUST** be set to the context identifier for BIB-HMAC-SHA2.
- Any local flags used to generate the IPPT **MUST** be placed in the integrity scope flags security parameter for the BIB unless these flags are expected to be correctly configured at security verifiers and acceptors in the network.
- The HMAC key **MAY** be included as a security parameter in which case it **MUST** be wrapped using the NIST AES-KW algorithm and the results of the wrapping added as the wrapped key security parameter for the BIB.
- The SHA variant used by this security context **SHOULD** be added as the SHA variant security parameter for the BIB if it differs from the default key length. Otherwise, this parameter **MAY** be omitted if doing so provides a useful reduction in message sizes.

Problems encountered in the keyed hash generation **MUST** be processed in accordance with local BPSec security policy.

3.8.2. Keyed Hash Verification

During keyed hash verification, the input of the security target and a HMAC key are provided to the appropriate HMAC/SHA2 algorithm.

During keyed hash verification, two inputs are prepared for the appropriate HMAC/SHA2 algorithm: the HMAC key and the IPPT. These data items **MUST** be generated as follows.

- The HMAC key **MUST** be derived using the wrapped key security parameter if such a parameter is included in the security context parameters of the BIB. Otherwise, this key **MUST** be derived in accordance with security policy at the verifying node as discussed in Section 3.5.
- The IPPT **MUST** be generated as discussed in Section 3.7 with the value of integrity scope flags being taken from the integrity scope flags security context parameter. If the integrity scope flags parameter is not included in the security context parameters then these flags **MAY** be derived from local security policy.

The calculated HMAC output **MUST** be compared to the expected HMAC output encoded in the security results of the BIB for the security target. If the calculated HMAC and expected HMAC are identical, the verification **MUST** be considered a success. Otherwise, the verification **MUST** be considered a failure.

If the verification fails or otherwise experiences an error, or if any needed parameters are missing, then the verification **MUST** be treated as failed and processed in accordance with local security policy.

This security service is removed from the bundle at the security acceptor as required by the BPSec specification. If the security acceptor is not the bundle destination and if no other integrity service is being applied to the target block, then a CRC **MUST** be included for the target block. The CRC type, as determined by policy, is set in the target block's CRC type field and the corresponding CRC value is added as the CRC field for that block.

4. Security Context BCB-AES-GCM

4.1. Overview

The BCB-AES-GCM security context replaces the block-type-specific data field of its security target with cipher text generated using the Advanced Encryption Standard (AES) cipher operating in Galois/Counter Mode (GCM) [AES-GCM]. The use of AES-GCM was selected as the cipher suite for this confidentiality mechanism for several reasons:

1. The selection of a symmetric-key cipher suite allows for relatively smaller keys than asymmetric-key cipher suites.
2. The selection of a symmetric-key cipher suite allows this security context to be used in places where an asymmetric-key infrastructure (such as a public key infrastructure) might be impractical.
3. The use of the Galois/Counter Mode produces cipher-text with the same size as the plain text making the replacement of target block information easier as length fields do not need to be changed.
4. The AES-GCM cipher suite provides authenticated encryption, as required by the BPSec protocol.

Additionally, the BCB-AES-GCM security context generates an authentication tag based on the plain text value of the block-type-specific data and other additional authenticated data that might be specified via parameters to this security context.

This security context supports two variants of AES-GCM, based on the supported length of the symmetric key. These variants correspond to A128GCM and A256GCM as defined in Table 9 ("Algorithm Value for AES-GCM") of [RFC8152].

The BCB-AES-GCM security context **MUST** have the security context identifier specified in Section 5.1.

4.2. Scope

There are two scopes associated with BCB-AES-GCM: the scope of the confidentiality service and the scope of the authentication service. The first defines the set of information provided to the AES-GCM cipher for the purpose of producing cipher text. The second defines the set of information used to generate an authentication tag.

The scope of the confidentiality service defines the set of information provided to the AES-GCM cipher for the purpose of producing cipher text. This **MUST** be the full set of plain text contained in the block-type-specific data field of the security target.

The scope of the authentication service defines the set of information used to generate an authentication tag carried with the security block. This information contains all data protected by the confidentiality service, the scope flags used to identify other optional information, and **MAY** include other information (additional authenticated data), as follows.

Primary block

The primary block identifies a bundle and, once created, the contents of this block are immutable. Changes to the primary block associated with the security target indicate that the security target (and BCB) might no longer be in the correct bundle.

For example, if a security target and associated BCB are copied from one bundle to another bundle, the BCB might still be able to decrypt the security target even though these blocks were never intended to exist in the copied-to bundle.

Including this information as part of additional authenticated data ensures that security target (and security block) appear in the same bundle at the time of decryption as at the time of encryption.

Security target other fields

The other fields of the security target include block identification and processing information. Changing this information changes how the security target is treated by nodes in the network even when the "user data" of the security target are otherwise unchanged.

For example, if the block processing control flags of a security target are different at a security verifier than they were originally set at the security source then the policy for handling the security target has been modified.

Including this information as part of additional authenticated data ensures that the cipher text in the security target will not be used with a different set of block policy than originally set at the time of encryption.

BCB other fields

The other fields of the BCB include block identification and processing information. Changing this information changes how the BCB is treated by nodes in the network, even when other aspects of the BCB are unchanged.

> For example, if the block processing control flags of the BCB are different at a security acceptor than they were originally set at the security source then the policy for handling the BCB has been modified.
>
> Including this information as part of additional authenticated data ensures that the policy and identification of the security service in the bundle has not changed.
>
> NOTE: The security context identifier and security context parameters of the security block are not included as additional authenticated data because these parameters, by definition, are those needed to verify or accept the security service. Therefore, it is expected that changes to these values would result in failures at security verifiers and security acceptors. This is the case because keys cannot be re-used across security contexts, and because the AAD scope flags used to identify the AAD are included in the AAD.

The scope of the BCB-AES-GCM security context is configured using an optional security context parameter.

4.3. Parameters

BCB-AES-GCM can be parameterized to specify the AES variant, initialization vector, key information, and identify additional authenticated data.

4.3.1. Initialization Vector (IV)

This optional parameter identifies the initialization vector (IV) used to initialize the AES-GCM cipher.

The length of the initialization vector, prior to any CBOR encoding, **MUST** be between 8-16 bytes. A value of 12 bytes **SHOULD** be used unless local security policy requires a different length.

This value **MUST** be encoded as a CBOR byte string.

The initialization vector can have any value with the caveat that a value **MUST NOT** be re-used for multiple encryptions using the same encryption key. This value **MAY** be re-used when encrypting with different keys. For example, if each encryption operation using BCB-AES-GCM uses a newly generated key, then the same IV can be reused.

4.3.2. AES Variant

This optional parameter identifies the AES variant being used for the AES-GCM encryption, where the variant is identified by the length of key used.

This value **MUST** be encoded as a CBOR unsigned integer.

Valid values for this parameter are as follows.

Value	Description
1	A128GCM as defined in Table 9 ("Algorithm Values for AES-GCM") of [RFC8152]

Value	Description
3	A256GCM as defined in Table 9 ("Algorithm Values for AES-GCM") of [RFC8152]

Table 4: AES Variant Parameter Values

When not provided, implementations **SHOULD** assume a value of 3 (indicating use of A256GCM), unless an alternate default is established by local security policy at the security source, verifier, or acceptor of this integrity service.

Regardless of the variant, the generated authentication tag **MUST** always be 128 bits.

4.3.3. Wrapped Key

This optional parameter contains the output of the AES key wrap authenticated encryption function (KW-AE) as defined in [RFC5649]. Specifically, this parameter holds the cipher text produced when running the KW-AE algorithm with the input string being the symmetric AES key used to generate the security results present in the security block. The value of this parameter is used as input to the AES key wrap authenticated decryption function (KW-AD) at security verifiers and security acceptors to determine the symmetric AES key needed for the proper decryption of the security results in the security block.

This value **MUST** be encoded as a CBOR byte string.

If this parameter is not present then security verifiers and acceptors **MUST** determine the proper key as a function of their local BPSec policy and configuration.

4.3.4. AAD Scope Flags

This optional parameter contains a series of flags that describe what information is to be included with the block-type-specific data of the security target as part of additional authenticated data (AAD).

This value **MUST** be represented as a CBOR unsigned integer, the value of which **MUST** be processed as a 16-bit field. The maximum value of this field, as a CBOR unsigned integer, **MUST** be 65535.

Implementations **MUST** set reserved and unassigned bits in this field to 0 when constructing these flags at a security source. Once set, the value of this field **MUST NOT** be altered until the security service is completed at the security acceptor in the network and removed from the bundle.

Bits in this field represent additional information to be included when generating an integrity signature over the security target. These bits are defined as follows.

Bit 0 (the low-order bit, 0x0001): Primary Block Flag

Bit 1 (0x0002): Target Header Flag

Bit 2 (0x0004): Security Header Flag

Bits 3-7: Reserved

Bits 8-15: Unassigned

4.3.5. Enumerations

The BCB-AES-GCM security context parameters are listed in Table 5. In this table, the "Parm Id" column refers to the expected Parameter Identifier described in Section 3.10 ("Parameter and Result Identification") of [RFCYYY2].

If the default value column is empty, this indicates that the security parameter does not have a default value.

Parm Id	Parm Name	CBOR Encoding Type	Default Value
1	Initialization Vector	Byte String	
2	AES Variant	Unsigned Integer	3
3	Wrapped Key	Byte String	
4	AAD Scope Flags	Unsigned Integer	7

Table 5: BCB-AES-GCM Security Parameters

4.4. Results

The BCB-AES-GCM security context produces a single security result carried in the security block: the authentication tag.

NOTES:

- The cipher text generated by the cipher suite is not considered a security result as it is stored in the block-type-specific data field of the security target block. When operating in GCM mode, AES produces cipher text of the same size as its plain text and, therefore, no additional logic is required to handle padding or overflow caused by the encryption in most cases (see below).
- If the authentication tag can be separated from the cipher text, then the tag **MAY** be separated and stored in the authentication tag security result field. Otherwise, the security target block **MUST** be resized to accommodate the additional 128 bits of authentication tag included with the generated cipher text replacing the block-type-specific-data field of the security target block.

4.4.1. Authentication Tag

The authentication tag is generated by the cipher suite over the security target plain text input to the cipher suite as combined with any optional additional authenticated data. This tag is used to ensure that the plain text (and important information associated with the plain text) is authenticated prior to decryption.

If the authentication tag is included in the cipher text placed in the security target block-type-specific data field, then this security result **MUST NOT** be included in the BCB for that security target.

The length of the authentication tag, prior to any CBOR encoding, **MUST** be 128 bits.

This value **MUST** be encoded as a CBOR byte string.

4.4.2. Enumerations

The BCB-AES-GCM security context results are listed in Table 6. In this table, the "Result Id" column refers to the expected Result Identifier described in Section 3.10 ("Parameter and Result Identification") of [RFCYYY2].

Result Id	Result Name	CBOR Encoding Type
1	Authentication Tag	Byte String

Table 6: BCB-AES-GCM Security Results

4.5. Key Considerations

Keys used with this context **MUST** be symmetric and **MUST** have a key length equal to the key length defined in the security context parameters or as defined by local security policy at security verifiers and acceptors. For this reason, content-encrypting key lengths will be integer divisible by 8 bytes and special padding-aware AES key wrap algorithms are not needed.

It is assumed that any security verifier or security acceptor can determine the proper key to be used. Potential sources of the key include (but are not limited to) the following.

- Pre-placed keys selected based on local policy.
- Keys extracted from material carried in the BCB.
- Session keys negotiated via a mechanism external to the BCB.

When an AES-KW wrapped key is present in a security block, it is assumed that security verifiers and security acceptors can independently determine the key encryption key (KEK) used in the wrapping of the symmetric AES content-encrypting key.

The security provided by block ciphers is reduced as more data is processed with the same key. The total number of AES blocks processed with a single key for AES-GCM is recommended to be less than 2^64, as described in Appendix B of [AES-GCM].

Additionally, there exist limits on the number of encryptions that can be performed with the same key. The total number of invocations of the authenticated encryption function with a single key for AES-GCM is required to not exceed 2^32, as described in Section 8.3 of [AES-GCM].

As discussed in Section 6 and emphasized here, it is strongly recommended that keys be protected once generated, both when they are stored and when they are transmitted.

4.6. GCM Considerations

The GCM cryptographic mode of AES has specific requirements that **MUST** be followed by implementers for the secure function of the BCB-AES-GCM security context. While these requirements are well documented in [AES-GCM], some of them are repeated here for emphasis.

- With the exception of the AES-KW function, the IVs used by the BCB-AES-GCM security context are considered to be per-invocation IVs. The pairing of a per-invocation IV and a security key **MUST** be unique. A per-invocation IV **MUST NOT** be used with a security key more than one time. If a per-invocation IV and key pair are repeated then the GCM implementation is vulnerable to forgery attacks. Because the loss of integrity protection occurs with even a single reuse, this situation is often considered to have catastrophic security consequences. More information regarding the importance of the uniqueness of the IV value can be found in Appendix A of [AES-GCM].

  Methods of generating unique IV values are provided in Chapter 8 of [AES-GCM]. For example, one method decomposes the IV value into a fixed field and an invocation field. The fixed field being a constant value associated with a device and the invocation field changing on each invocation (such as by incrementing an integer counter). Implementers **SHOULD** carefully read all relevant sections of [AES-GCM] when generating any mechanism to create unique IVs.
- The AES-KW function used to wrap keys for the security contexts in this document uses a single, globally constant IV input to the AES cipher operation and, thus, is distinct from the aforementioned requirement related to per-invocation IVs.
- While any tag-based authentication mechanism has some likelihood of being forged, this probability is increased when using AES-GCM. In particular, short tag lengths combined with very long messages **SHOULD** be avoided when using this mode. The BCB-AES-GCM security context requires the use of 128-bit authentication tags at all times. Concerns relating to the size of authentication tags is discussed in Appendices B and C of [AES-GCM].
- As discussed in Appendix B of [AES-GCM], implementations **SHOULD** limit the number of unsuccessful verification attempts for each key to reduce the likelihood of guessing tag values. This type of check has potential state-keeping issues when AES-KW is used, since an attacker could cause a large number of keys to have been used at least once.
- As discussed in Section 8 ("Security Considerations") of [RFCYYY2], delay-tolerant networks have a higher occurrence of replay attacks due to the store-and-forward nature of the network. Because GCM has no inherent replay attack protection, implementors **SHOULD** attempt to detect replay attacks by using mechanisms such as those described in Appendix D of [AES-GCM].

4.7. Canonicalization Algorithms

This section defines the canonicalization algorithms used to prepare the inputs used to generate both the cipher text and the authentication tag.

In all cases, the canonical form of any portion of an extension block **MUST** be performed as described in [RFCYYY2]. The canonicalization algorithms defined in [RFCYYY2] adhere to the canonical forms for extension blocks defined in [RFCYYY1] but resolve ambiguities related to how values are represented in CBOR.

4.7.1. Cipher text related calculations

The BCB operates over the block-type-specific data of a block, but the BP always encodes these data within a single, definite-length CBOR byte string. Therefore, the plain text used during encryption **MUST** be calculated as the value of the block-type-specific data field of the security target excluding the BP CBOR encoding.

Consider the following two CBOR encoded examples and the plain text that would be extracted from them. The first example is an unsigned integer, while the second is a byte string.

CBOR Encoding (Hex)	CBOR Part (Hex)	Plain Text Part (Hex)
18ED	18	ED
C24CDEADBEEFDEADBEEFDEADBEEF	C24C	DEADBEEFDEADBEEFDEADBEEF

Table 7: CBOR Plain Text Extraction Examples

Similarly, the cipher text used during decryption **MUST** be calculated as the single, definite-length CBOR byte string representing the block-type-specific data field excluding the CBOR byte string identifying byte and optional CBOR byte string length field.

All other fields of the security target (such as the block type code, block number, block processing control flags, or any CRC information) **MUST NOT** be considered as part of encryption or decryption.

4.7.2. Additional Authenticated Data

The construction of additional authenticated data depends on the AAD scope flags that can be provided as part of customizing the behavior of this security context.

The canonical form of the AAD input to the BCB-AES-GCM mechanism is constructed using the following process. While the AAD scope flags might not be included in the BCB representing the security operation, they **MUST** be included in the AAD value itself. This process **MUST** be followed when generating AAD for either encryption or decryption.

1. The canonical form of the AAD starts as the CBOR encoding of the AAD scope flags in which all unset flags, reserved bits, and unassigned bits have been set to 0. For example, if the primary block flag, target header flag, and security header flag are each set, then the initial value of the canonical form of the AAD will be 0x07.
2. If the primary block flag of the AAD scope flags is set to 1, then a canonical form of the bundle's primary block **MUST** be calculated and the result appended to the AAD.

3. If the target header flag of the AAD scope flags is set to 1, then the canonical form of the block type code, block number, and block processing control flags associated with the security target **MUST** be calculated and, in that order, appended to the AAD.
4. If the security header flag of the AAD scope flags is set to 1, then the canonical form of the block type code, block number, and block processing control flags associated with the BIB **MUST** be calculated and, in that order, appended to the AAD.

4.8. Processing

4.8.1. Encryption

During encryption, four inputs are prepared for input to the AES/GCM cipher: the encryption key, the IV, the security target plain text to be encrypted, and any additional authenticated data. These data items **MUST** be generated as follows.

Prior to encryption, if a CRC value is present for the target block, then that CRC value **MUST** be removed. This requires removing the CRC field from the target block and setting the CRC type field of the target block to "no CRC is present."

- The encryption key **MUST** have the appropriate length as required by local security policy. The key might be generated specifically for this encryption, given as part of local security policy, or through some other key management mechanism as discussed in Section 4.5.
- The IV selected **MUST** be of the appropriate length. Because replaying an IV in counter mode voids the confidentiality of all messages encrypted with said IV, this context also requires a unique IV for every encryption performed with the same key. This means the same key and IV combination **MUST NOT** be used more than once.
- The security target plain text for encryption **MUST** be generated as discussed in Section 4.7.1.
- Additional authenticated data **MUST** be generated as discussed in Section 4.7.2 with the value of AAD scope flags being taken from local security policy.

Upon successful encryption the following actions **MUST** occur.

- The cipher text produced by AES/GCM **MUST** replace the bytes used to define the plain text in the security target block's block-type-specific data field. The block length of the security target **MUST** be updated if the generated cipher text is larger than the plain text (which can occur when the authentication tag is included in the cipher text calculation, as discussed in Section 4.4).
- The authentication tag calculated by the AES/GCM cipher **MAY** be added as a security result for the security target in the BCB holding results for this security operation, in which case it **MUST** be processed as described in Section 4.4.
- The authentication tag **MUST** be included either as a security result in the BCB representing the security operation or (with the cipher text) in the security target block-type-specific data field.

Finally, the BCB containing information about this security operation **MUST** be updated as follows. These operations can occur in any order.

- The security context identifier for the BCB **MUST** be set to the context identifier for BCB-AES-GCM.
- The IV input to the cipher **MUST** be added as the IV security parameter for the BCB.
- Any local flags used to generated AAD for this cipher **MUST** be placed in the AAD scope flags security parameter for the BCB unless these flags are expected to be correctly configured at security verifiers and security acceptors in the network.
- The encryption key **MAY** be included as a security parameter in which case it **MUST** be wrapped using the NIST AES-KW algorithm and the results of the wrapping added as the wrapped key security parameter for the BCB.
- The AES variant used by this security context **SHOULD** be added as the AES variant security parameter for the BCB if it differs from the default key length. Otherwise, this parameter **MAY** be omitted if doing so provides a useful reduction in message sizes.

Problems encountered in the encryption **MUST** be processed in accordance with local security policy. This **MAY** include restoring a CRC value removed from the target block prior to encryption, if the target block is allowed to be transmitted after an encryption error.

4.8.2. Decryption

During decryption, five inputs are prepared for input to the AES/GCM cipher: the decryption key, the IV, the security target cipher text to be decrypted, any additional authenticated data, and the authentication tag generated from the original encryption. These data items **MUST** be generated as follows.

- The decryption key **MUST** be derived using the wrapped key security parameter if such a parameter is included in the security context parameters of the BCB. Otherwise this key **MUST** be derived in accordance with local security policy at the decrypting node as discussed in Section 4.5.
- The IV **MUST** be set to the value of the IV security parameter included in the BCB. If the IV parameter is not included as a security parameter, an IV **MAY** be derived as a function of local security policy and other BCB contents or a lack of an IV security parameter in the BCB **MAY** be treated as an error by the decrypting node.
- The security target cipher text for decryption **MUST** be generated as discussed in Section 4.7.1.
- Additional authenticated data **MUST** be generated as discussed in Section 4.7.2 with the value of AAD scope flags being taken from the AAD scope flags security context parameter. If the AAD scope flags parameter is not included in the security context parameters then these flags **MAY** be derived from local security policy in cases where the set of such flags is determinable in the network.
- The authentication tag **MUST** be present either as a security result in the BCB representing the security operation or (with the cipher text) in the security target block-type-specific data field.

Upon successful decryption the following actions **MUST** occur.

- The plain text produced by AES/GCM **MUST** replace the bytes used to define the cipher text in the security target block's block-type-specific data field. Any changes to the security target block length field **MUST** be corrected in cases where the plain text has a different length than the replaced cipher text.

If the security acceptor is not the bundle destination and if no other integrity or confidentiality service is being applied to the target block, then a CRC **MUST** be included for the target block. The CRC type, as determined by policy, is set in the target block's CRC type field and the corresponding CRC value is added as the CRC field for that block.

If the cipher text fails to authenticate, if any needed parameters are missing, or if there are other problems in the decryption then the decryption **MUST** be treated as failed and processed in accordance with local security policy.

5. IANA Considerations

5.1. Security Context Identifiers

This specification allocates two security context identifiers from the "BPSec Security Context Identifiers" registry defined in [RFCYYY2].

Value	Description	Reference
TBA	BIB-HMAC-SHA2	This document
TBA	BCB-AES-GCM	This document

Table 8: Additional Entries for the BPSec Security Context Identifiers Registry

5.2. Integrity Scope Flags

The BIB-HMAC-SHA2 security context has an Integrity Scope Flags field for which IANA is requested to create and maintain a new registry named "BPSec BIB-HMAC-SHA2 Integrity Scope Flags" on the Bundle Protocol registry page. Initial values for this registry are given below.

The registration policy for this registry is: Specification Required.

The value range is unsigned 16-bit integer.

Bit Position (right to left)	Description	Reference
0	Include primary block	This document
1	Include target header flag	This document

Bit Position (right to left)	Description	Reference
2	Include security header flag	This document
3-7	reserved	This document
8-15	unassigned	This document

Table 9: BPSec BIB-HMAC-SHA2 Integrity Scope Flags Registry

5.3. AAD Scope Flags

The BCB-AES-GCM security context has an AAD Scope Flags field for which IANA is requested to create and maintain a new registry named "BPSec BCB-AES-GCM AAD Scope Flags" on the Bundle Protocol registry page. Initial values for this registry are given below.

The registration policy for this registry is: Specification Required.

The value range is unsigned 16-bit integer.

Bit Position (right to left)	Description	Reference
0	Include primary block	This document
1	Include target header flag	This document
2	Include security header flag	This document
3-7	reserved	This document
8-15	unassigned	This document

Table 10: BPSec BCB-AES-GCM AAD Scope Flags Registry

5.4. Guidance for Designated Experts

New assignments within the BIB-HMAC-SHA2 Integrity Scope Flags Registry and the BCB-AES-GCM AAD Scope Flags Registry require review by a Designated Expert (DE). This section provides guidance to the DE when performing their reviews. Specifically, a DE is expected to perform the following activities.

- Ascertain the existence of suitable documentation (a specification) as described in [RFC8126] and to verify that the document is permanently and publicly available.
- Ensure that any changes to the Integrity Scope Flags clearly state how new assignments interact with existing flags and how the inclusion of new assignments affects the construction of the IPPT value.
- Ensure that any changes to the AAD Scope Flags clearly state how new assignments interact with existing flags and how the inclusion of new assignments affects the construction of the AAD input to the BCB-AES-GCM mechanism.

- Ensure that any processing changes proposed with new assignments do not alter any required behavior in this specification.

6. Security Considerations

Security considerations specific to a single security context are provided in the description of that context. This section discusses security considerations that should be evaluated by implementers of any security context described in this document. Considerations can also be found in documents listed as normative references and they should also be reviewed by security context implementors.

6.1. Key Management

The delayed and disrupted nature of DTNs complicates the process of key management because there might not be reliable, timely round-trip exchange between security sources, security verifiers, and security acceptors in the network. This is true when there is a substantial signal propagation delay between nodes, when nodes are in a highly challenged communications environment, and when nodes do not support bi-directional communication.

In these environments, key establishment protocols that rely on round-trip information exchange might not converge on a shared secret in a timely manner (or at all). Also, key revocation or key verification mechanisms that rely on access to a centralized authority (such as a certificate authority) might similarly fail in the stressing conditions of a DTN.

For these reasons, the default security contexts described in this document rely on symmetric key cryptographic mechanisms because asymmetric key infrastructure (such as a public key infrastructure) might be impractical in this environment.

BPSec assumes that "key management is handled as a separate part of network management" [RFCYYY2]. This assumption is also made by the security contexts defined in this document which do not define new protocols for key derivation, exchange of key-encrypting keys, revocation of existing keys, or the security configuration or policy used to select certain keys for certain security operations.

Nodes using these security contexts need to perform the following kinds of activities, independent of the construction, transmission, and processing of BPSec security blocks.

- Establish shared key-encrypting-keys with other nodes in the network using an out-of-band mechanism. This might include pre-sharing of key encryption keys or the use of traditional key establishment mechanisms prior to the exchange of BPsec security blocks.
- Determine when a key is considered exhausted and no longer to be used in the generation, verification, or acceptance of a security block.
- Determine when a key is considered invalid and no longer to be used in the generation, verification, or acceptance of a security block. Such revocations can be based on a variety of mechanisms to include local security policy, time relative to the generation or use of the key, or as specified through network management.

- Determine, through an out-of-band mechanism such as local security policy, what keys are to be used for what security blocks. This includes the selection of which key should be used in the evaluation of a security block received by a security verifier or a security acceptor.

The failure to provide effective key management techniques appropriate for the operational networking environment can result in the compromise of those unmanaged keys and the loss of security services in the network.

6.2. Key Handling

Once generated, keys should be handled as follows.

- It is strongly **RECOMMENDED** that implementations protect keys both when they are stored and when they are transmitted.
- In the event that a key is compromised, any security operations using a security context associated with that key **SHOULD** also be considered compromised. This means that the BIB-HMAC-SHA2 security context **SHOULD NOT** be treated as providing integrity when used with a compromised key and BCB-AES-GCM **SHOULD NOT** be treated as providing confidentiality when used with a compromised key.
- The same key, whether a key-encrypting-key or a wrapped key, **MUST NOT** be used for different algorithms as doing so might leak information about the key.
- A key-encrypting-key **MUST NOT** be used to encrypt keys for different security contexts. Any key-encrypting-key used by a security context defined in this document **MUST** only be used to wrap keys associated with security operations using that security context. This means that a compliant security source would not use the same key-encrypting-key to wrap keys for both the BIB-HMAC-SHA2 and BCB-AES-GCM security contexts. Similarly, any compliant security verifier or security acceptor would not use the same key-encrypting-key to unwrap keys for different security contexts.

6.3. AES GCM

There are a significant number of considerations related to the use of the GCM mode of AES to provide a confidentiality service. These considerations are provided in Section 4.6 as part of the documentation of the BCB-AES-GCM security context.

The length of the cipher text produced by the GCM mode of AES will be equal to the length of the plain text input to the cipher suite. The authentication tag also produced by this cipher suite is separate from the cipher text. However, it should be noted that implementations of the AES-GCM cipher suite might not separate the concept of cipher text and authentication tag in their application programming interface (API).

Implementations of the BCB-AES-GCM security context can either keep the length of the target block unchanged by holding the authentication tag in a BCB security result or alter the length of the target block by including the authentication tag with the cipher text replacing the block-type-specific-data field of the target block. Implementations **MAY** use the authentication tag security result in cases where keeping target block length unchanged is an important processing concern. In all cases, the cipher text and authentication tag **MUST** be processed in accordance with the API of the AES-GCM cipher suites at the security source and security acceptor.

6.4. AES Key Wrap

The AES key wrap (AES-KW) algorithm used by the security contexts in this document does not use a per-invocation initialization vector and does not require any key padding. Key padding is not needed because wrapped keys used by these security contexts will always be multiples of 8 bytes. The length of the wrapped key can be determined by inspecting the security context parameters. Therefore, a key can be unwrapped using only the information present in the security block and the key encryption key provided by local security policy at the security verifier or security acceptor.

6.5. Bundle Fragmentation

Bundle fragmentation might prevent security services in a bundle from being verified after a bundle is fragmented and before the bundle is re-assembled. Examples of potential issues include the following.

- If a security block and its security target do not exist in the same fragment, then the security block cannot be processed until the bundle is re-assembled. If a fragment includes an encrypted target block, but not its BCB, then a receiving bundle processing agent (BPA) will not know that the target block has been encrypted.
- A security block can be cryptographically bound to a bundle by setting the Integrity Scope Flags (for BIB-HMAC-SHA2) or the AAD Scope Flags (for BCB-AES-GCM) to include the bundle primary block. When a security block is cryptographically bound to a bundle, it cannot be processed even if the security block and target both coexist in the fragment. This is because fragments have different primary blocks than the original bundle.
- If security blocks and their target blocks are repeated in multiple fragments, policy needs to determine how to deal with issues where a security operation verifies in one fragment but fails in another fragment. This might happen, for example, if a BIB block becomes corrupted in one fragment but not in another fragment.

Implementors should consider how security blocks are processed when a BPA fragments a received bundle. For example, security blocks and their targets could be placed in the same fragment if the security block is not otherwise cryptographically bound to the bundle being fragmented. Alternatively, if security blocks are cryptographically bound to a bundle, then a fragmenting BPA should consider encapsulating the bundle first and then fragmenting the encapsulating bundle.

7. Normative References

[AES-GCM]

Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST Special Publication 800-38D, DOI 10.6028/NIST.SP.800-38D, November 2007, <https://doi.org/10.6028/NIST.SP.800-38D>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5649] Housley, R. and M. Dworkin, "Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm", RFC 5649, DOI 10.17487/RFC5649, September 2009, <https://www.rfc-editor.org/info/rfc5649>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8742] Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", RFC 8742, DOI 10.17487/RFC8742, February 2020, <https://www.rfc-editor.org/info/rfc8742>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[RFCYYY1] Burleigh, S., Fall, K., and E. Birrane, "Bundle Protocol Version 7", RFC YYY1, DOI 10.17487/RFCYYY1, September 2021, <https://www.rfc-editor.org/rfc/rfcYYY1>.

[RFCYYY2] Birrane, E. and K. McKeever, "Bundle Protocol Security Specification", RFC YYY2, DOI 10.17487/RFCYYY2, September 2021, <https://www.rfc-editor.org/rfc/rfcYYY2>.

[SHS] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, DOI 10.6028/NIST.FIPS.180-4, August 2015, <https://csrc.nist.gov/publications/detail/fips/180/4/final>.

Appendix A. Examples

This appendix is informative.

This section presents a series of examples of constructing BPSec security blocks (using the security contexts defined in this document) and adding those blocks to a sample bundle.

The examples presented in this appendix represent valid constructions of bundles, security blocks, and the encoding of security context parameters and results. For this reason, they can inform unit test suites for individual implementations as well as interoperability test suites amongst implementations. However, these examples do not cover every permutation of security parameters, security results, or use of security blocks in a bundle.

NOTE: The bundle diagrams in this section are patterned after the bundle diagrams used in Section 3.11 ("BSP Block Examples") of [RFCYYY2].

NOTE: Figures in this section identified as "(CBOR Diagnostic Notation)" are represented using the CBOR diagnostic notation defined in [RFC8949]. This notation is used to express CBOR data structures in a manner that enables visual inspection. The bundles, security blocks, and security context contents in these figures are represented using CBOR structures. In cases where BP blocks (to include BPSec security blocks) are comprised of a sequence of CBOR objects, these objects are represented as a CBOR sequence as defined in [RFC8742].

NOTE: Examples in this section use the "ipn" URI scheme for EndpointID naming, as defined in [RFCYYY1].

NOTE: The bundle source is presumed to be the security source for all security blocks in this section, unless otherwise noted.

A.1. Example 1: Simple Integrity

This example shows the addition of a BIB to a sample bundle to provide integrity for the payload block.

A.1.1. Original Bundle

The following diagram shows the original bundle before the BIB has been added.

Block in Bundle	Block Type	Block Number
Primary Block	N/A	0
Payload Block	1	1

Table 11: Example 1: Original Bundle

A.1.1.1. Primary Block

The BPv7 bundle has no special processing flags and no CRC is provided because the primary block is expected to be protected by an integrity service BIB using the BIB-HMAC-SHA2 security context.

The bundle is sourced at the source node ipn:2.1 and destined for the destination node ipn:1.2. The bundle creation time uses a DTN creation time of 0 indicating lack of an accurate clock and a sequence number of 40. The lifetime of the bundle is given as 1,000,000 milliseconds since the bundle creation time.

The primary block is provided as follows.

```
[
  7,           / BP version            /
  0,           / flags                 /
  0,           / CRC type              /
  [2, [1,2]],  / destination (ipn:1.2) /
  [2, [2,1]],  / source      (ipn:2.1) /
  [2, [2,1]],  / report-to   (ipn:2.1) /
  [0, 40],     / timestamp             /
  1000000      / lifetime              /
]
```

Figure 1: Primary Block (CBOR Diagnostic Notation)

The CBOR encoding of the primary block is 0x88070000820282010282028202018202820201820018281a000f4240.

A.1.1.2. Payload Block

Other than its use as a source of plaintext for security blocks, the payload has no required distinguishing characteristic for the purpose of this example. The sample payload is a 32 byte string whose value is "Ready Generate a 32 byte payload".

The payload is represented in the payload block as a byte string of the raw payload string. It is NOT represented as a CBOR text string wrapped within a CBOR binary string. The hex value of the payload "Ready Generate a 32 byte payload" is 0x52656164792047656e657261746520612033322062797465207061796c6f6164.

The payload block is provided as follows.

```
[
  1,      / type code: Payload block /
  1,      / block number             /
  0,      / block processing flags   /
  0,      / CRC Type                 /
  h'52656164792047656e65726174652061 / type-specific-data: payload /
    2033322062797465207061796c6f6164'
]
```

Figure 2: Payload Block (CBOR Diagnostic Notation)

The CBOR encoding of the payload block is 0x8501010000582052656164792047656e657261746520612033322062797465207061796c6f6164.

A.1.1.3. Bundle CBOR Representation

A BPv7 bundle is represented as an indefinite-length array consisting of the blocks comprising the bundle, with a terminator character at the end.

The CBOR encoding of the original bundle is 0x9f88070000820282010282028202018202820201820018281a000f4240850101000058205265616479204765 6e6572

A.1.2. Security Operation Overview

This example adds a BIB to the bundle using the BIB-HMAC-SHA2 security context to provide an integrity mechanism over the payload block.

The following diagram shows the resulting bundle after the BIB is added.

Block in Bundle	Block Type	Block Number
Primary Block	N/A	0
Bundle Integrity Block OP(bib-integrity, target=1)	11	2
Payload Block	1	1

Table 12: Example 1: Resulting Bundle

A.1.3. Bundle Integrity Block

In this example, a BIB is used to carry an integrity signature over the payload block.

A.1.3.1. Configuration, Parameters, and Results

For this example, the following configuration and security parameters are used to generate the security results indicated.

This BIB has a single target and includes a single security result: the calculated signature over the payload block.

```
                  Key         : h'1a2b1a2b1a2b1a2b1a2b1a2b1a2b1a2b'
                  SHA Variant : HMAC 512/512
                  Scope Flags : 0x00
                  Payload Data: h'52656164792047656e65726174652061
                                  2033322062797465207061796c6f6164'
                  Signature   : h'0654d65992803252210e377d66d0a8dc
                                  18a1e8a392269125ae9ac198a9a598be
                                  4b83d5daa8be2f2d16769ec1c30cfc34
                                  8e2205fba4b3be2b219074fdd5ea8ef0'
```

Figure 3: Example 1: Configuration, Parameters, and Results

A.1.3.2. Abstract Security Block

The abstract security block structure of the BIB's block-type-specific-data field for this application is as follows.

```
[1],            / Security Target        - Payload block       /
1,              / Security Context ID    - BIB-HMAC-SHA2       /
1,              / Security Context Flags - Parameters Present  /
[2,[2, 1]],     / Security Source        - ipn:2.1             /
[               / Security Parameters    - 2 Parameters        /
   [1, 7],      / SHA Variant            - HMAC 512/512        /
   [3, 0x00]    / Scope Flags            - No Additional Scope /
],
[           / Security Results: 1 Result /
   [1, h'0654d65992803252210e377d66d0a8dc18a1e8a392269125ae9ac198a9a598b
   e4b83d5daa8be2f2d16769ec1c30cfc348e2205fba4b3be2b219074fdd5ea8ef0']
]
```

Figure 4: Example 1: BIB Abstract Security Block (CBOR Diagnostic Notation)

The CBOR encoding of the BIB block-type-specific-data field (the abstract security block) is 0x810101018202820201828201078203008182015840 0654d65992803252210e377d66d0a8dc18a1e8a392269125ae

A.1.3.3. Representations

The BIB wrapping this abstract security block is as follows.

```
[
  11, / type code    /
  2,  / block number /
  0,  / flags        /
  0,  / CRC type     /
  h'810101018202820201828201078203008182015840 0654d65992803252210e377d66
  d0a8dc18a1e8a392269125ae9ac198a9a598be4b83d5daa8be2f2d16769ec1c30cfc34
  8e2205fba4b3be2b219074fdd5ea8ef0',
]
```

Figure 5: Example 1: BIB (CBOR Diagnostic Notation)

The CBOR encoding of the BIB block is 0x850b0200005855810101018202820201828201078203008182015840 0654d65992803252210e377d66d0a8dc18a1

A.1.4. Final Bundle

The CBOR encoding of the full output bundle, with the BIB: 0x9f88070000820282010282028202018202820201820018281a000f4240850b0200005855810101018202820201828

A.2. Example 2: Simple Confidentiality with Key Wrap

This example shows the addition of a BCB to a sample bundle to provide confidentiality for the payload block. AES key wrap is used to transmit the symmetric key used to generate the security results for this service.

A.2.1. Original Bundle

The following diagram shows the original bundle before the BCB has been added.

Block in Bundle	Block Type	Block Number
Primary Block	N/A	0
Payload Block	1	1

Table 13: Example 2: Original Bundle

A.2.1.1. Primary Block

The primary block used in this example is identical to the primary block presented in Example 1 Appendix A.1.1.1.

In summary, the CBOR encoding of the primary block is 0x88070000820282010282028202018202820201820018281a000f4240.

A.2.1.2. Payload Block

The payload block used in this example is identical to the payload block presented in Example 1 Appendix A.1.1.2.

In summary, the CBOR encoding of the payload block is 0x85010100005820526561647920746f2067656e657261746520612033322062797465207061796c6f6164.

A.2.1.3. Bundle CBOR Representation

A BPv7 bundle is represented as an indefinite-length array consisting of the blocks comprising the bundle, with a terminator character at the end.

The CBOR encoding of the original bundle is 0x9f88070000820282010282028202018202820201820018281a000f424085010100005820526561647920746f2067656e657

A.2.2. Security Operation Overview

This example adds a BCB using the BCB-AES-GCM security context using AES key wrap to provide a confidentiality mechanism over the payload block and transmit the symmetric key.

The following diagram shows the resulting bundle after the BCB is added.

Block in Bundle	Block Type	Block Number
Primary Block	N/A	0
Bundle Confidentiality Block OP(bcb-confidentiality, target=1)	12	2
Payload Block (Encrypted)	1	1

Table 14: Example 2: Resulting Bundle

A.2.3. Bundle Confidentiality Block

In this example, a BCB is used to encrypt the payload block and uses AES key wrap to transmit the symmetric key.

A.2.3.1. Configuration, Parameters, and Results

For this example, the following configuration and security parameters are used to generate the security results indicated.

This BCB has a single target, the payload block. Three security results are generated: cipher text which replaces the plain text block-type-specific data to encrypt the payload block, an authentication tag, and the AES wrapped key.

```
Content Encryption
               Key: h'71776572747975696f70617364666768'
Key Encryption Key: h'6162636465666768696a6b6c6d6e6f70'
                IV: h'5477656c7665313231323132'
       AES Variant: A128GCM
   AES Wrapped Key: h'69c411276fecddc4780df42c8a2af892
                      96fabf34d7fae700'
       Scope Flags: 0x00
      Payload Data: h'52656164792047656e65726174652061
                      2033322062797465207061796c6f6164'
Authentication Tag: h'da08f4d8936024ad7c6b3b800e73dd97'
Payload Ciphertext: h'3a09c1e63fe2097528a78b7c12943354
                      a563e32648b700c2784e26a990d91f9d'
```

Figure 6: Example 2: Configuration, Parameters, and Results

A.2.3.2. Abstract Security Block

The abstract security block structure of the BCB's block-type-specific-data field for this application is as follows.

```
[1],                / Security Target        - Payload block       /
2,                  / Security Context ID    - BCB-AES-GCM         /
1,                  / Security Context Flags - Parameters Present  /
[2,[2, 1]],         / Security Source        - ipn:2.1             /
[                   / Security Parameters    - 4 Parameters        /
  [1, h'5477656c7665313231323132'], / Initialization Vector       /
  [2, 1],                           / AES Variant - A128GCM       /
  [3, h'69c411276fecddc4780df42c8a  / AES wrapped key             /
        2af89296fabf34d7fae700'],
  [4, 0x00]                         / Scope Flags - No extra scope/
],
[                                   /  Security Results: 1 Result /
  [1, h'da08f4d8936024ad7c6b3b800e73dd97']    / Payload Auth. Tag /
]
```

Figure 7: Example 2: BCB Abstract Security Block (CBOR Diagnostic Notation)

The CBOR encoding of the BCB block-type-specific-data field (the abstract security block) is 0x8101020182028202018482014c5477656c76653132313231328202018203581869c411276fecddc4780df42c8a2af8

A.2.3.3. Representations

The BCB wrapping this abstract security block is as follows.

```
[
  12, / type code /
  2,  / block number /
  1,  / flags - block must be replicated in every fragment /
  0,  / CRC type /
  h'8101020182028202018482014c5477656c766531323132313282020182035818
    69c411276fecddc4780df42c8a2af89296fabf34d7fae70082040081820150da
    08f4d8936024ad7c6b3b800e73dd97'
]
```

Figure 8: Example 2: BCB (CBOR Diagnostic Notation)

The CBOR encoding of the BCB block is 0x850c020100584f8101020182028202018482014c5477656c76653132313231328202018203581869c411276fecddc4

A.2.4. Final Bundle

The CBOR encoding of the full output bundle, with the BCB: 0x9f88070000820282010282028202018202820201820018281a000f4240850c020100584f8101020182028202018482

A.3. Example 3: Security Blocks from Multiple Sources

This example shows the addition of a BIB and BCB to a sample bundle. These two security blocks are added by two different nodes. The BCB is added by the source endpoint and the BIB is added by a forwarding node.

The resulting bundle contains a BCB to encrypt the Payload Block and a BIB to provide integrity to the Primary and Bundle Age Block.

A.3.1. Original Bundle

The following diagram shows the original bundle before the security blocks have been added.

Block in Bundle	Block Type	Block Number
Primary Block	N/A	0
Extension Block: Bundle Age Block	7	2
Payload Block	1	1

Table 15: Example 3: Original Bundle

A.3.1.1. Primary Block

The primary block used in this example is identical to the primary block presented in Example 1 Appendix A.1.1.1.

In summary, the CBOR encoding of the primary block is 0x88070000820282010282028202018202820201820018281a000f4240.

A.3.1.2. Bundle Age Block

A bundle age block is added to the bundle to help other nodes in the network determine the age of the bundle. The use of this block is as recommended because the bundle source does not have an accurate clock (as indicated by the DTN time of 0).

Because this block is specified at the time the bundle is being forwarded, the bundle age represents the time that has elapsed from the time the bundle was created to the time it is being prepared for forwarding. In this case, the value is given as 300 milliseconds.

The bundle age extension block is provided as follows.

```
[
  7,      / type code: Bundle Age block /
  2,      / block number /
  0,      / block processing flags /
  0,      / CRC Type /
  <<300>> / type-specific-data: age /
]
```

Figure 9: Bundle Age Block (CBOR Diagnostic Notation)

The CBOR encoding of the bundle age block is 0x85070200004319012c.

A.3.1.3. Payload Block

The payload block used in this example is identical to the payload block presented in Example 1 Appendix A.1.1.2.

In summary, the CBOR encoding of the payload block is 0x85010100005820526561647920476f6e657261746520612033322062797465207061796c6f6164.

A.3.1.4. Bundle CBOR Representation

A BPv7 bundle is represented as an indefinite-length array consisting of the blocks comprising the bundle, with a terminator character at the end.

The CBOR encoding of the original bundle is 0x9f88070000820282010282028202018202820201820018281a000f424085070200004319012c850101000058205265

A.3.2. Security Operation Overview

This example provides:

- a BIB with the BIB-HMAC-SHA2 security context to provide an integrity mechanism over the primary block and bundle age block.
- a BCB with the BCB-AES-GCM security context to provide a confidentiality mechanism over the payload block.

The following diagram shows the resulting bundle after the security blocks are added.

Block in Bundle	Block Type	Block Number
Primary Block	N/A	0
Bundle Integrity Block OP(bib-integrity, targets=0, 2)	11	3
Bundle Confidentiality Block OP(bcb-confidentiality, target=1)	12	4
Extension Block: Bundle Age Block	7	2
Payload Block (Encrypted)	1	1

Table 16: Example 3: Resulting Bundle

A.3.3. Bundle Integrity Block

In this example, a BIB is used to carry an integrity signature over the bundle age block and an additional signature over the payload block. The BIB is added by a waypoint node, ipn:3.0.

A.3.3.1. Configuration, Parameters, and Results

For this example, the following configuration and security parameters are used to generate the security results indicated.

This BIB has two security targets and includes two security results, holding the calculated signatures over the bundle age block and primary block.

```
                 Key: h'1a2b1a2b1a2b1a2b1a2b1a2b1a2b1a2b'
         SHA Variant: HMAC 256/256
         Scope Flags: 0x00
  Primary Block Data: h'88070000820282010282028202018202
                        820201820018281a000f4240'
  Bundle Age Block
                Data: h'85070200004319012c'
  Primary Block
           Signature: h'8e059b8e71f7218264185a666bf3e453
                        076f2b883f4dce9b3cdb6464ed0dcf0f'
  Bundle Age Block
           Signature: h'72dee8eba049a22978e84a95d0496466
                        8eb131b1ca4800c114206d70d9065c80'
```

Figure 10: Example 3: Configuration, Parameters, and Results for the BIB

A.3.3.2. Abstract Security Block

The abstract security block structure of the BIB's block-type-specific-data field for this application is as follows.

```
[0, 2],         / Security Targets                           /
1,              / Security Context ID    - BIB-HMAC-SHA2       /
1,              / Security Context Flags - Parameters Present  /
[2,[3, 0]],     / Security Source        - ipn:3.0             /
[               / Security Parameters    - 2 Parameters        /
   [1, 5],      / SHA Variant            - HMAC 256/256        /
   [3, 0x00]    / Scope Flags            - No Additional Scope /
],
[           / Security Results: 2 Results /
   [1, h'8e059b8e71f7218264185a666bf3e453
         076f2b883f4dce9b3cdb6464ed0dcf0f'], / Primary Block   /
   [1, h'72dee8eba049a22978e84a95d0496466
         8eb131b1ca4800c114206d70d9065c80'] / Bundle Age Block /
]
```

Figure 11: Example 3: BIB Abstract Security Block (CBOR Diagnostic Notation)

The CBOR encoding of the BIB block-type-specific-data field (the abstract security block) is 0x820002010182028203008282010582030082820158208e059b8e71f7218264185a666bf3e453076f2b883f4dce9b3c

A.3.3.3. Representations

The BIB wrapping this abstract security block is as follows.

```
[
  11, / type code /
  3,  / block number /
  0,  / flags  /
  0,  / CRC type /
  h'820002010182028203008282010582030082820158208e059b8e71f721826418
    5a666bf3e453076f2b883f4dce9b3cdb6464ed0dcf0f8201582072dee8eba049
    a22978e84a95d04964668eb131b1ca4800c114206d70d9065c80',
]
```

Figure 12: Example 3: BIB (CBOR Diagnostic Notation)

The CBOR encoding of the BIB block is 0x850b030000585a820002010182028203008282010582030082820158208e059b8e71f7218264185a666bf3e453076

A.3.4. Bundle Confidentiality Block

In this example, a BCB is used encrypt the payload block. The BCB is added by the bundle source node, ipn:2.1.

A.3.4.1. Configuration, Parameters, and Results

For this example, the following configuration and security parameters are used to generate the security results indicated.

This BCB has a single target, the payload block. Two security results are generated: cipher text which replaces the plain text block-type-specific data to encrypt the payload block, and an authentication tag.

```
Content Encryption
               Key: h'71776572747975696f70617364666768'
                IV: h'5477656c7665313231323132'
       AES Variant: A128GCM
       Scope Flags: 0x00
      Payload Data: h'526561647920476f656e65726174652061
                      2033322062797465207061796c6f6164'
Authentication Tag: h'da08f4d8936024ad7c6b3b800e73dd97'
Payload Ciphertext: h'3a09c1e63fe2097528a78b7c12943354
                      a563e32648b700c2784e26a990d91f9d'
```

Figure 13: Example 3: Configuration, Parameters, and Results for the BCB

A.3.4.2. Abstract Security Block

The abstract security block structure of the BCB's block-type-specific-data field for this application is as follows.

```
[1],             / Security Target        - Payload block      /
2,               / Security Context ID    - BCB-AES-GCM        /
1,               / Security Context Flags - Parameters Present /
[2,[2, 1]],      / Security Source        - ipn:2.1            /
[                / Security Parameters    - 3 Parameters       /
  [1, h'5477656c7665313231323132'],    / Initialization Vector /
  [2, 1],                              / AES Variant - AES 128 /
  [4, 0x00]                / Scope Flags - No Additional Scope /
],
[                                     / Security Results: 1 Result /
  [1, h'da08f4d8936024ad7c6b3b800e73dd97'] / Payload Auth. Tag /
]
```

Figure 14: Example 3: BCB Abstract Security Block (CBOR Diagnostic Notation)

The CBOR encoding of the BCB block-type-specific-data field (the abstract security block) is 0x8101020182028202018382014c5477656c76653132313231328202018204008182015 0da08f4d8936024ad7c6b3b8

A.3.4.3. Representations

The BCB wrapping this abstract security block is as follows.

```
[
  12, / type code /
  4,  / block number /
  1,  / flags - block must be replicated in every fragment /
  0,  / CRC type /
  h'8101020182028202018382014c5477656c766531323132313282020182040081
    820150da08f4d8936024ad7c6b3b800e73dd97',
]
```

Figure 15: Example 3: BCB (CBOR Diagnostic Notation)

The CBOR encoding of the BCB block is 0x850c04010058338101020182028202018382014c5477656c7665313231323132820201820400818201 50da08f4d89

A.3.5. Final Bundle

The CBOR encoding of the full output bundle, with the BIB and BCB added is: 0x9f88070000820282010282028202018202820201820018281a000f4240850b030000585a8200020101820282030 08

A.4. Example 4: Security Blocks with Full Scope

This example shows the addition of a BIB and BCB to a sample bundle. A BIB is added to provide integrity over the payload block and a BCB is added for confidentiality over the payload and BIB.

The integrity scope and additional authentication data will bind the primary block, target header, and the security header.

A.4.1. Original Bundle

The following diagram shows the original bundle before the security blocks have been added.

Block in Bundle	Block Type	Block Number
Primary Block	N/A	0
Payload Block	1	1

Table 17: Example 4: Original Bundle

A.4.1.1. Primary Block

The primary block used in this example is identical to the primary block presented in Example 1 Appendix A.1.1.1.

In summary, the CBOR encoding of the primary block is 0x88070000820282010282028202018202820201820018281a000f4240.

A.4.1.2. Payload Block

The payload block used in this example is identical to the payload block presented in Example 1 Appendix A.1.1.2.

In summary, the CBOR encoding of the payload block is 0x8501010000582052656164792047656e657261746520612033322062797465207061796c6f6164.

A.4.1.3. Bundle CBOR Representation

A BPv7 bundle is represented as an indefinite-length array consisting of the blocks comprising the bundle, with a terminator character at the end.

The CBOR encoding of the original bundle is 0x9f88070000820282010282028202018202820201820018281a000f42408501010000582052656164792047656e657

A.4.2. Security Operation Overview

This example provides:

- a BIB with the BIB-HMAC-SHA2 security context to provide an integrity mechanism over the payload block.
- a BCB with the BCB-AES-GCM security context to provide a confidentiality mechanism over the payload block and BIB.

The following diagram shows the resulting bundle after the security blocks are added.

Block in Bundle	Block Type	Block Number
Primary Block	N/A	0

Block in Bundle	Block Type	Block Number
Bundle Integrity Block (Encrypted) OP(bib-integrity, target=1)	11	3
Bundle Confidentiality Block OP(bcb-confidentiality, targets=1, 3)	12	2
Payload Block (Encrypted)	1	1

Table 18: Example 4: Resulting Bundle

A.4.3. Bundle Integrity Block

In this example, a BIB is used to carry an integrity signature over the payload block. The IPPT contains the payload block block-type-specific data, primary block data, the payload block header, and the BIB header. That is, all additional headers are included in the IPPT.

A.4.3.1. Configuration, Parameters, and Results

For this example, the following configuration and security parameters are used to generate the security results indicated.

This BIB has a single target and includes a single security result: the calculated signature over the Payload block.

```
                Key: h'1a2b1a2b1a2b1a2b1a2b1a2b1a2b1a2b'
        SHA Variant: HMAC 384/384
        Scope Flags: 0x07  (all additional headers)
 Primary Block Data: h'88070000820282010282028202018202
                       820201820018281a000f4240
       Payload Data: h'52656164792047656e65726174652061
                       2033322062797465207061796c6f6164'
     Payload Header: h'85010100005820'
         BIB Header: h'850b0300005845'
  Payload Signature: h'07c84d929f83bee4690130729d77a1bd
                       da9611cd6598e73d0659073ea74e8c27
                       523b02193cb8ba64be58dbc556887aca
```

Figure 16: Example 4: Configuration, Parameters, and Results for the BIB

A.4.3.2. Abstract Security Block

The abstract security block structure of the BIB's block-type-specific-data field for this application is as follows.

```
[1],            / Security Target            - Payload block           /
1,              / Security Context ID        - BIB-HMAC-SHA2           /
1,              / Security Context Flags     - Parameters Present      /
[2,[2, 1]],     / Security Source            - ipn:2.1                 /
[               / Security Parameters        - 2 Parameters            /
   [1, 6],      / SHA Variant                - HMAC 384/384            /
   [3, 0x07]    / Scope Flags - All additional headers in the SHA Hash /
],
[               / Security Results: 1 Result /
   [1, h'07c84d929f83bee4690130729d77a1bdda9611cd6598e73d
         0659073ea74e8c27523b02193cb8ba64be58dbc556887aca']
]
```

Figure 17: Example 4: BIB Abstract Security Block (CBOR Diagnostic Notation)

The CBOR encoding of the BIB block-type-specific-data field (the abstract security block) is 0x81010101820282020182820106820307818201583007c84d929f83bee4690130729d77a1bdda9611cd6598e73d065

A.4.3.3. Representations

The BIB wrapping this abstract security block is as follows.

```
[
  11, / type code /
  3,  / block number /
  0,  / flags  /
  0,  / CRC type /
  h'81010101820282020182820106820307818201583007c84d929f83bee4690130
    729d77a1bdda9611cd6598e73d0659073ea74e8c27523b02193cb8ba64be58db
    c556887aca',
]
```

Figure 18: Example 4: BIB (CBOR Diagnostic Notation)

The CBOR encoding of the BIB block is 0x850b030000584581010101820282020182820106820307818201583007c84d929f83bee4690130729d77a1bdda961

A.4.4. Bundle Confidentiality Block

In this example, a BCB is used encrypt the payload block and the BIB that provides integrity over the payload.

A.4.4.1. Configuration, Parameters, and Results

For this example, the following configuration and security parameters are used to generate the security results indicated.

This BCB has two targets: the payload block and BIB. Four security results are generated: cipher text which replaces the plain text block-type-specific data of the payload block, cipher text to encrypt the BIB, and authentication tags for both the payload block and BIB.

Key:

```
h'71776572747975696f70617364666768
  71776572747975696f70617364666768'
```

IV:

```
h'5477656c7665313231323132'
```

AES Variant:
A256GCM

Scope Flags:
0x07 (All additional headers)

Payload Data:

```
h'526561647920476656e6572617465206120
  2033322062797465207061796c6f6164'
```

BIB Data:

```
h'810101018202820201828201068203 07
  818201583007c84d929f83bee4690130
  729d77a1bdda9611cd6598e73d065907
  3ea74e8c27523b02193cb8ba64be58db
  c556887aca
```

BIB Authentication Tag:

```
h'c95ed4534769b046d716e1cdfd00830e'
```

Payload Block Authentication Tag:

```
h'0e365c700e4bb19c0d991faff5345aff'
```

Payload Ciphertext:

```
h'90eab64575930498d6aa654107f15e96
  319bb227706000abc8fcac3b9bb9c87e'
```

BIB Ciphertext:

```
h'438ed6208eb1c1ffb94d952175167df0
  902a815f221ebc837a134efc13bfa82a
  2d5d317747da3eb54acef4ca839bd961
  487284404259b60be12b8aed2f3e8a36
  2836529f66'
```

A.4.4.2. Abstract Security Block

The abstract security block structure of the BCB's block-type-specific-data field for this application is as follows.

```
[3, 1],           / Security Targets                           /
2,                / Security Context ID    - BCB-AES-GCM        /
1,                / Security Context Flags - Parameters Present /
[2,[2, 1]],       / Security Source        - ipn:2.1            /
[                 / Security Parameters    - 3 Parameters       /
  [1, h'5477656c7665313231323132'],    / Initialization Vector /
  [2, 3],                              / AES Variant - AES 256 /
  [4, 0x07]            / Scope Flags - All headers in SHA hash /
],
[                                 / Security Results: 2 Results /
  [1, h'c95ed4534769b046d716e1cdfd00830e'],    / BIB Auth. Tag /
  [1, h'0e365c700e4bb19c0d991faff5345aff'] / Payload Auth. Tag /
]
```

Figure 19: Example 4: BCB Abstract Security Block (CBOR Diagnostic Notation)

The CBOR encoding of the BCB block-type-specific-data field (the abstract security block) is 0x82030102018202820201838201 4c5477656c7665313231323132820203820407828201 50c95ed4534769b046d716e

A.4.4.3. Representations

The BCB wrapping this abstract security block is as follows.

```
[
  12, / type code /
  2,  / block number /
  1,  / flags - block must be replicated in every fragment /
  0,  / CRC type /
  h'820301020182028202018382014c5477656c766531323132313282020382040 7
    82820150c95ed4534769b046d716e1cdfd00830e8201500e365c700e4bb19c0d
    991faff5345aff',
]
```

Figure 20: Example 4: BCB (CBOR Diagnostic Notation)

The CBOR encoding of the BCB block is 0x850c0201005847820301020182028202018382014c5477656c7665313231323132820203820407828201 50c95ed45

A.4.5. Final Bundle

The CBOR encoding of the full output bundle, with the security blocks added and payload block and BIB encrypted is:
0x9f88070000820282010282028202018202820201820018281a000f4240850b0300005845438ed6208eb1c1ffb94d95
850c0201005847820301020182028202018382014c5477656c76653132313231328202038204078282015 0c95ed4534

Appendix B. CDDL Expression

For informational purposes, Brian Sipos has kindly provided an expression of the IPPT and AAD structures using the Concise Data Definition Language (CDDL). That CDDL expression is presented below.

Note that wherever the CDDL expression is in disagreement with the textual representation of the security block specification presented in earlier sections of this document, the textual representation rules.

Note that the structure of BP bundles and BPSec security blocks are provided by other specifications and this section only provides the CDDL expression for structures uniquely defined in this specification. Items related to elements of a bundle, such as "primary-block", are defined in Appendix B of the Bundle Protocol Version 7 [RFCYYY1].

Note that the CDDL itself does not have the concept of unadorned CBOR sequences as a top-level subject of a specification. The current best practice, as documented in Section 4.1 of [RFC8742], requires representing the sequence as an array with a comment in the CDDL noting that the array represents a CBOR sequence.

```
start = scope / AAD-list / IPPT-list ; satisfy CDDL decoders

scope = uint .bits scope-flags
scope-flags = &(
    has-primary-ctx: 0,
    has-target-ctx: 1,
    has-security-ctx: 2,
)

; Encoded as a CBOR sequence
AAD-list = [
    AAD-structure
]

; Encoded as a CBOR sequence
IPPT-list = [
    AAD-structure,
    target-btsd: bstr ; block-type-specific-data of the target block.
]

AAD-structure = (
    scope,
    ? primary-block,  ; present if has-primary-ctx flag set
    ? block-metadata, ; present if has-target-ctx flag set
    ? block-metadata, ; present if has-security-ctx flag set
)

; Selected fields of a canonical block
block-metadata = (
    block-type-code: uint,
    block-number: uint,
    block-control-flags,
)
```

Figure 21: IPPT and AAD Expressions

Appendix C. Acknowledgements

Amy Alford of the Johns Hopkins University Applied Physics Laboratory contributed useful review and analysis of these security contexts.

Authors' Addresses

Edward J. Birrane, III
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
United States of America
Phone: +1 443 778 7423
Email: Edward.Birrane@jhuapl.edu

Alex White
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
United States of America
Phone: +1 443 778 0845
Email: Alex.White@jhuapl.edu

Sarah Heiner
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
United States of America
Phone: +1 240 592 3704
Email: Sarah.Heiner@jhuapl.edu